Blockwise simple component analysis via rotation, constraints or penalties, with an application to product × attribute × panelist data

Kiers, Henk A. L.; Timmerman, Marieke E.; Ceulemans, Eva

Published in:
Food Quality and Preference

DOI:
10.1016/j.foodqual.2017.01.018

Publication date:
2018

Blockwise Simple Component Analysis via Rotation, Constraints or Penalties, with an application to product × attribute ×panelist data

Henk A.L. Kiers[a]

Marieke E. Timmerman[b]

Eva Ceulemans[b]

[a]) Heymans Institute, University of Groningen, Grote Kruisstraat 2/1, 9712TS Groningen, The Netherlands; h.a.l.kiers@rug.nl.
[b]) Heymans Institute, University of Groningen, Grote Kruisstraat 2/1, 9712TS Groningen, The Netherlands; m.e.timmerman@rug.nl.
[c]) Quantitative Psychology and Individual Differences, KU Leuven, Tiensestraat 102, B-3000 Leuven, Belgium; Eva.Ceulemans@ kuleuven.be.

Corresponding author: Henk A.L. Kiers, Heymans Institute, University of Groningen, Grote Kruisstraat 2/1, 9712TS Groningen, The Netherlands; h.a.l.kiers@rug.nl.

Abstract

Sensory profiling data consisting of judgements on a number of products with respect to a number of attributes by a number of panelists can be summarized in various ways. Besides finding components describing the main product features, there is an interest in individual panelist behavior. Earlier methods identify this by means of separate PCAs, Procrustes analyses, or three-way component methods, but these give only global comparisons of panelists. In the present paper, methods that can distinguish panelist behavior related to *separate* attributes, are described. These methods model the data in such a way that blocks of loadings pertaining to the attributes are either small or large. At the same time, one can zoom in on the loadings for panelists within each block of loadings associated with an attribute to inspect differences in panelist behavior. Two types of methods have been proposed for this earlier (rotation to simple blocks and penalizing blocks of loadings), and a third one is proposed in the present paper (constraining blocks of loadings to zero). The new approach is compared here to the other two methods. It is found that the rotation and constraints approaches work about equally well and better than the penalty approach. However, the rotation approach offers richer panelist behavior information, as is illustrated by the analysis of empirical data. It is also shown how, in this example, the reliability of idiosyncratic panelist behavior indicators can be evaluated.

Keywords: Sparse Group Component Analysis, multiset data, simple structure rotation, sensory profiling data

Highlights:

- Three methods for summarizing sensory profile data are compared
- These methods focus both on product characteristics and panelist behaviour
- Sparse Group Component Analysis performed relatively poorly here
- Blockwise Simplimax and Blockwise Zero Constrained CA performed well
- Blockwise Simplimax also identifies idiosyncratic panelist behaviour

1 Introduction

Conventional sensory profiling involves judgements on a number of products, with respect to a number of attributes, by a number of panelists. The resulting three-way data are to be analyzed to identify the main product features, and possibly the individual differences between panelists. If identifying the product features is the ultimate goal, one might be tempted to average the judgements across panelists, and then identify the product features by means of a Principal Components Analysis (PCA) on the averages. However, such a summary ignores possible important individual differences between panelists, and may be biased because of that. Therefore, it is always important to take individual differences between panelists into account.

Various methods have been proposed for analyzing sensory profiling data. Dijksterhuis and Punter (1990) proposed Generalized Procrustes analysis on the products × attributes matrices to assess per panelist to what extent it agrees with the others across all attributes. Dijksterhuis (1995) proposed to use separate Principal components analyses on the products × panelists matrices for each of the attributes. For each attribute, the proportion of explained variance by the first component is taken as a measure of agreement among panelists on the attribute at hand. Both methods lead only to rather global conclusions, and do not identify precisely what special behavior some panelists display. Qannari, Wakeling, & MacFie (1995) and Bro, Qannari, Kiers, Naes and Frøst (2008) go beyond that in their proposal to use STATIS (Lavit, 1988) and Parafac (Harshman, 1970), respectively to assess individual panelists behavior. Wilderjans and Cariou (2016) follow up on this by proposing to use Clustered Parafac (Krijnen & Kiers, 1993), so as to get easier interpretable solutions. These methods can be very successful for the purpose of summarizing product features and panelist behavior. However, because these are three-way methods, the description of panelist behavior is necessarily related to summarizing *components* for the attributes, and not to individual attributes. Therefore, when a panelist treats a single attribute differently compared to how the other panelists treat it, this cannot be identified by a three-way method (compare De Roover, Timmerman, Van Mechelen, & Ceulemans, 2013, who discuss the limitations of three-way methods).

To identify such deviant panelist behavior, one needs to consider the relations of each panelist × attribute combination with the product features components. In two-way methods, such as a Principal Component Analysis (PCA) applied to the matrix with as variables all panelist ×attribute combinations, these relations are represented (i.e., by

means of the loadings). Therefore, here, like in De Roover et al. (2013), we consider two-way component methods to analyze three-way data.

One of the main criticisms against analyzing three-way data by two-way methods is that two-way methods fail to identify the three-way structure in the data, as three-way component analysis, that is, Parafac (Harshman, 1970) or 3MPCA (Tucker, 1966; Kroonenberg & De Leeuw, 1980), does. The latter gives a concise model for the data, which makes the interpretation of the components relatively easy. That is, in a three-way component analysis, the interpretation of components for the attributes only involves the loadings of the attributes, while in a two-way component analysis, the components are related to *all* panelist × attribute combinations. Suppose one has 23 attributes and 8 panelists (as in the example data set analyzed in Section 4 of this paper). Then, in a three-way analysis, loadings for the attribute components deal with only 23 attributes, while in the two-way analysis, one has to deal with 184 variables related to the panelist × attribute combinations. Hence, the advantage of having the latter set of loadings available for identifying deviant panelist behavior, is a disadvantage at the same time, in that the presence of so many loadings seriously hinders the interpretation of the components. The methods we discuss here, deal with this dilemma by interpreting the components in a blockwise manner, and by aiming to simplify the loadings in a blockwise manner. That is, the methods discussed here aim at finding blocks of loadings (each block being associated with one attribute) that are either small or large, thus allowing for an easy *global* interpretation of the components in terms of the attributes. At the same time, deviant individual panelist behavior can also be revealed, by inspecting the loadings of the panelists within blocks of loadings, and searching for salient discrepancies.

For the goal of finding blocks that are either small or large, two methods have been proposed in the literature. The first consists of PCA (or more precisely, estimating a Tucker1 model (Tucker 1966, Kroonenberg & de Leeuw, 1980)) followed by Blockwise Simplimax, as proposed by Timmerman, Kiers, & Ceulemans (2016). Blockwise Simplimax involves rotating the loadings such that blocks of loadings have either small or large loadings. The second consists of a form of penalized PCA, as proposed by Van Deun et al. (2011). This method aims for zero loadings for a number of blocks, while the remaining loadings can be expected to be relatively large. In this paper, a third approach is used, as in Adachi and Trendafilov's (2016) sparseness constrained PCA. Specifically we constrain a number of blocks of loadings to be zero, while the others are again

expected to be large. Timmerman et al. compared the first two methods, and found the first to be favorable in most conditions of their simulation study, and never to perform considerably worse. The third blockwise approach, to our knowledge, is new, and will be compared to the first two methods in the present paper. In addition, a second simulation study, with data sizes more in line with sensory data, is carried out to compare the constrained methods to the rotation method. This is the first goal of our paper.

As Timmerman et al. (2016) mentioned, a particularly interesting feature of PCA followed by Blockwise Simplimax, is that it does not only give an easy way for global interpretation of a large data set, but it can also be used to identify deviant panelist behavior. By using the same empirical data set as Timmerman et al. do, we will demonstrate more specifically what this entails, and propose a method for assessing the reliability of the identification of such behavior. Furthermore, in a second simulation study we will test to what extent the method actually identifies such behavior. This is the second goal of the paper.

The paper starts with a description of the methods in Section 2. In Section 3, the methods will be compared on the basis of a simulation study, and in Section 4, the analysis of the empirical data set is described. The paper concludes with a discussion in Section 5. The paper can in various ways be considered as a follow up on the Timmerman et al. (2016) paper, using the same design of the simulation study therein, as well as the same empirical data set. The essence of results and set-ups there is repeated here, but for the full picture, it is advised to consult that paper as well.

2 Blockwise simple component methods

To describe Blockwise simple component methods for three-way data, first some generally used notation will be given. We assume that the data consist of scores for I products on K attributes, given by J panelists. For each attribute, the data are collected in an $I \times J$ data block, denoted as $\mathbf{X}_k$ for attribute k. The full data set is given by the $(I \times JK)$ matrix $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_K)$. The data are typically preprocessed by centering across products (which is here assumed to be the case throughout), and some kind of normalization, for instance such that each column or each block has unit sum of squares. Centering across products implies that scores will be considered relative to the mean product score (for each attribute × panelist combination separately). Normalization such that for all attribute blocks the scores have the same sum of squares, ensures that all attributes are given the same variability, and hence are treated as equally important. Normalization

within columns implies that this holds for all attribute × panelist combinations. Which choice is to be preferred depends on substantive considerations.
In all methods, the PCA model, or a constrained version thereof, is fitted. The unconstrained PCA model can be described as

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{T}} + \mathbf{E}, \qquad (1)$$

with $\mathbf{T}$ ($I \times R$) containing the component scores of the I products on R components, $\mathbf{P}$ ($JK \times R$) containing the loadings, and $\mathbf{E}$ the ($I \times JK$) matrix with residuals. The component score matrix $\mathbf{T}$ is here required to be columnwise orthonormal, i.e., $\mathbf{T}^{\mathrm{T}}\mathbf{T}=\mathbf{I}$. As a consequence, the loadings are covariances (and correlations if the data are normalized per column) between components and attribute × panelist combinations, which is convenient when interpreting components.

2.1 PCA followed by Blockwise Simplimax rotation

In unconstrained PCA, model (1) is fitted to the data by minimizing the loss function

$$f_{\mathrm{PCA}}(\mathbf{T},\mathbf{P}) = \| \mathbf{X} - \mathbf{T}\mathbf{P}^{\mathrm{T}} \|^2, \qquad (2)$$

subject to the constraint $\mathbf{T}^{\mathrm{T}}\mathbf{T}=\mathbf{I}$. It is well known that the model has 'rotational freedom' meaning that for any orthonormal rotation matrix $\mathbf{R}$ (i.e., for which $\mathbf{R}^{\mathrm{T}}\mathbf{R}= \mathbf{R}\mathbf{R}^{\mathrm{T}}=\mathbf{I}$), the component score matrix $\mathbf{T}^*=\mathbf{T}\mathbf{R}$ and the loading matrix $\mathbf{P}^*=\mathbf{P}\mathbf{R}$ give the same fit. This is because $\mathbf{T}^*(\mathbf{P}^*)^{\mathrm{T}} = \mathbf{T}\mathbf{R}(\mathbf{P}\mathbf{R})^{\mathrm{T}} = \mathbf{T}\mathbf{R}\mathbf{R}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}} = \mathbf{T}\mathbf{P}^{\mathrm{T}}$, from which $f_{\mathrm{PCA}}(\mathbf{T}^*,\mathbf{P}^*) = f_{\mathrm{PCA}}(\mathbf{T},\mathbf{P})$ follows at once. The rotational freedom is often exploited to rotate the loadings such that they become either small or large, and hence the components become easier to interpret. Many methods for such simple structure rotation have been proposed. Among the most popular procedures for this is Normalized Varimax (Kaiser, 1958), which aims at high variances of squared loadings per component. Normalized Varimax thus aims at finding both small and large loadings per component. An alternative, which gives more freedom as to the location of the small versus large loadings, is Simplimax rotation (Kiers, 1994). Specifically, Simplimax rotates the loadings such that the smallest p loadings have the smallest possible sum of squares, where p is a number to be specified in advance. Timmerman et al. (2016) proposed a blockwise variant of Simplimax. Their proposal is to rotate the loadings such that the smallest *p blockwise* sums of squares of loadings have the smallest possible sum. To this end, they minimize

$$g(\mathbf{W}, \mathbf{R}) = \sum_{k=1}^{K} \sum_{r=1}^{R} (1 - w_{kr}) \|\mathbf{p}^*_{\boldsymbol{kr}}\|^2 \ , \qquad (3)$$

where **W** (*K*×*R*) is a binary weight matrix with *p* zeros and *KR*-*p* unit elements, $\mathbf{p}_{kr}$ denotes the block of *J* loadings of all panelist × attribute combinations for attribute *k* on component *r* in the rotated loading matrix **P***=**PR**. Function *g* thus specifies the sum of squared loadings for all blocks for which w_{kr}=0. Because the method minimizes *g* not only over **R** but also over **W** (i.e., over which *p* values of **W** are to be made 0), the method actually seeks the rotation such that the *smallest* blockwise sum of squared loadings is found[1].

Function *g* is minimized by a variant of the Simplimax algorithm. This is an iterative algorithm, the outcome of which depends quite strongly on the binary weight matrix with which the procedure is started. It is therefore advised to use many differently started runs of the algorithm (e.g., 100), and find the best solution from these. The starts for the binary matrix can be random binary matrices (with *p* zeros), or binary matrices based on, for instance, the Varimax rotation of the loadings (i.e. setting the *p* blocks in **W** that correspond to the blocks with smallest sums of squares in the Varimax rotated loading matrix to zero). Next, the algorithm alternately updates **T,** keeping **W** fixed, and **W** keeping **T** fixed, until convergence, see Timmerman et al. (2016).

The method requires specifying *p* in advance. In practice, it is advised to obtain the optimal rotation for a large number of different values of *p* and to choose among these upon comparing the associated values of *g* for the different values of *p*. When we increase *p*, the loss function value *g* in (3) must increase or remain equal, because the sum of the *p+1* smallest possible values $||\mathbf{p}^*_{kr}||^2$ is always at least as large as that of the *p* smallest such values. In order to select *p,* it is then recommended to order all values *g* and their associated *p*'s from low to high and choose that value of *p after* which the value of *g* increases rapidly, which can be seen as a bend in a plot of *g* against *p*. Alternatively, one could apply the CHull procedure (Wilderjans, Ceulemans & Meers, 2013), a heuristic to identify the bend. The CHULL method is a general method for offsetting fit (or loss) against complexity (or simplicity) of a model. It consists of first determining the convex hull for the points in a fit versus complexity plot, and next computes the point(s) where the biggest ratio(s) of subsequent increases occurs. Solutions with such big ratios are

[1] Timmerman et al. also cover the more general case where the (number of) panelists differ per attribute. For that case they used a weighting of $||\mathbf{p}^*_{kr}||^2$ in (3) by J_k^{-1}, in order to avoid that the larger blocks influence the solution disproportionally.

considered as the most interesting solutions, because they indicate that, after such a solution, the fit increase is relatively small.

To give an idea of what such a rotated loading matrix and the associated **W** may look like, we analyzed a small subset of the cheeses data set collected by Frøst (2002, also see Bro et al. 2008), which will be described and analyzed in full in Section 4. Specifically, we selected the scores of all 30 products on four variables (M-Firm, M-Melt down, M-Fat and M-Butter), and considered only the first four panelists. We applied PCA to these data (after centering across products and normalizing the blocks to sum of squares 1), and found that the first four components accounted cumulatively for 44.4%, 71.8%, 78.0% and 82.2%. On the basis of a scree test, 2 components are clearly indicated. The resulting loading matrix is given in Table 1 (3rd and 4th column). Next, we applied Blockwise Simplimax to this loading matrix, for p=1,…,7. It was found that the resulting sums of squares of smallest blocks of loadings increased considerably after p=4, so we took p=4 in this example. The resulting block structure (as indicated by matrix **W**) and rotated Blockwise Simplimax loadings are given in the 5th-8th columns of Table 1. It can easily be seen that the blocks of smallest loading sums of squares (made grey) indeed contain small loadings in general.

Table 1. Small Example of applying Blockwise Simplimax (blocks of small loadings made grey), Sparse Group CA and Blockwise Zero Constrained CA

		Unrotated PCA loadings		**Blockwise Simplimax W**		**Blockwise Simplimax loadings**		**Sparse Group CA**		**Blockwise Zero Constrained CA**	
Attribute	Panelist	1	2	1	2	1	2	1	2	1	2
M-Firm	1	.26	-.18	**1**	**0**	.32	-.02	.20	0	.31	0
	2	.32	-.23			.40	-.01	.26	0	.40	0
	3	.43	-.23			.48	-.08	.32	0	.48	0
	4	.46	-.29			.54	-.05	.36	0	.55	0
M-Melt down	1	-.25	.24	**1**	**0**	-.35	-.04	-.21	0	-.34	0
	2	-.24	.00			-.19	.14	-.12	0	-.19	0
	3	-.41	.23			-.47	.07	-.29	0	-.48	0
	4	-.45	.25			-.51	.08	-.32	0	-.51	0
M-Fat	1	-.47	-.28	**0**	**1**	-.20	.51	0	.31	0	.51
	2	-.33	-.19			-.15	.35	0	.22	0	.36
	3	-.06	-.17			.05	.17	0	.11	0	.17
	4	-.20	-.42			.09	.45	0	.27	0	.45
M-Butter	1	-.31	-.25	**0**	**1**	-.10	.39	0	.25	0	.39
	2	-.33	-.25			-.11	.40	0	.26	0	.40
	3	-.28	-.25			-.07	.37	0	.23	0	.37
	4	-.24	-.45			.08	.50	0	.31	0	.49

2.2 Sparse Group Component Analysis (Sparse Group CA)

Rather than rotating loadings such that they have blockwise simple structure, one may also penalize loadings such that they become blockwise small. A disadvantage of this might be that by penalizing the loss function, the fit of the data will be reduced. However, a good data fit may in part be caused by overfitting the data, and hence does not necessarily imply better recovery of the underlying structure. Which method is to be preferred, therefore, is an empirical question, which can be answered by using for instance cross-validation techniques. In this paper, we use as a penalizing approach the "Group Lasso" within the framework of penalized component methods proposed by Van Deun et al. (2011). This method comes down to minimizing

$$f_{\text{SGCA}}(\mathbf{T},\mathbf{P}) = ||\mathbf{X} - \mathbf{T}\mathbf{P}^{\text{T}}||^2 + \lambda \sum_{r=1}^{R} \sum_{k=1}^{K} \|\mathbf{p}_{kr}\|, \quad (4)$$

subject to $\mathbf{T}^{\text{T}}\mathbf{T}=\mathbf{I}$, where λ denotes the so-called penalty parameter, and the term after it is the penalty term. An iterative algorithm for minimizing (4) has been given by Van Deun et al. The penalty term resembles function (3), but features an important difference in that not the squared norm, but the norm of the loadings in block (k,r) is taken. Minimizing the loss function including this penalty has the effect that for large enough λ, one or more blocks $\mathbf{p}_{kr}$ become exactly 0. This property resembles that of the Lasso penalty on individual parameters, hence the name Group Lasso. Thus, for sufficiently large λ, Sparse Group CA can be expected to yield component loadings that are either 0 or relatively large. In practice, one has to find a suitable value of λ by comparing various solutions with different λ. No concrete advice is available on how to choose λ, but one can, for instance, increase λ gradually until a well interpretable solution is found.

We also applied Sparse Group CA to the example data set described in Section 2.2. We first chose the penalty parameters in accordance with the suggestions by Van Deun et al, but most of these choices led to no zero blocks at all. Therefore, we further increased the penalty parameters. We found one block of zeros for λ=.10, three for λ=.25, and four for λ=.30, λ=.35, etc. (for all values checked). At the same time, thus increasing λ we did increase the loss considerably, because, due to the penalty, the loadings generally shrink. Therefore, we chose to consider the best solution among those with four zero blocks, that is for λ=.30. The loadings are displayed in Table 1 (9$^{\text{th}}$ and 10$^{\text{th}}$ columns). It can be seen that the structure of the Sparse Group CA is similar to that of the Blockwise Simplimax, but now with zero values instead of small values. Further, it is

salient that the non-zero loadings are considerably smaller than those from Blockwise Simplimax[2]. Obviously, due to the zero blocks, the Sparse Group CA solution is simpler than the one based on Blockwise Simplimax rotation. In general, with rotation one cannot expect to get blocks with exact zero loadings. In practice, the overall blockwise interpretation with respect to the attributes would be the same, since the blocks of small loadings will be contrasted to the blocks of relatively large loadings.

2.3 Blockwise Zero Constrained Component Analysis

In Section 2.2 it has been seen that Sparse Group CA, for sufficiently large values of λ, will give blocks of zero loadings. A straightforward alternative that directly searches blocks of zero loadings, is to simply constrain the loadings matrix **P** to have a number (say p) of blocks of zero loadings. This is the objective of the new method proposed here and called "Blockwise Zero Constrained Component Analysis", or Blockwise Zero Constrained CA for short. It thus minimizes

$$f_{\text{BZCCA}}(\mathbf{T},\mathbf{P}) = ||\,\mathbf{X} - \mathbf{T}\mathbf{P}^{\text{T}}\,||^2, \tag{5}$$

subject to $\mathbf{T}^{\text{T}}\mathbf{T}=\mathbf{I}$ and to the constraint that p blocks $\mathbf{p}_{kr}$ are exactly 0.

To solve this minimization problem, it is useful to recognize it as a generalization of Adachi and Trendafilov's (2016) method for PCA with the constraint that p loadings are zero. Their method thus covers the special case of our method in which each block has only a single row. Analogous to their approach, we propose an alternating least squares algorithm for updating **T** and **P**. For given **P**, the optimal **T** is given by $\mathbf{T} = \mathbf{U}\mathbf{V}^{\text{T}}$, where **U** and **V** are taken from the singular value decomposition $\mathbf{X}\mathbf{P} = \mathbf{U}\mathbf{D}\mathbf{V}^{\text{T}}$. For given **T**, the problem of optimizing **P** boils down to minimizing $||\,\mathbf{X} - \mathbf{T}\mathbf{P}^{\text{T}}\,||^2 = \text{tr}\,\mathbf{X}^{\text{T}}\mathbf{X} - 2\text{tr}\mathbf{X}^{\text{T}}\mathbf{T}\mathbf{P}^{\text{T}} + \text{tr}\mathbf{P}\mathbf{P}^{\text{T}} = \text{tr}\,\mathbf{X}^{\text{T}}\mathbf{X} - \text{tr}\,\mathbf{X}^{\text{T}}\mathbf{T}\mathbf{T}^{\text{T}}\mathbf{X} + ||\,\mathbf{X}^{\text{T}}\mathbf{T} - \mathbf{P}\,||^2$. As the first two terms are constant for **P**, the remaining problem is to find the matrix **P**, that minimizes $||\,\mathbf{X}^{\text{T}}\mathbf{T} - \mathbf{P}\,||^2$ subject to the constraint that p blocks $\mathbf{p}_{kr}$ are exactly 0. Upon defining $\mathbf{A} = \mathbf{X}^{\text{T}}\mathbf{T}$, with blocks $\mathbf{a}_{kr}$, one can see that the minimum of this function is found by setting $\mathbf{p}_{kr} = \mathbf{0}$ for the p blocks corresponding to the p smallest values of $||\mathbf{a}_{kr}||^2$, while $\mathbf{p}_{kr} = \mathbf{a}_{kr}$ for the other KR-p blocks. Thus alternatingly updating **T** and **P** monotonically decreases f_{BZCCA} and will converge to a stable value. The solution from a single run depends on the starting values. It may lead to the global minimum of the loss function, but in practice it often leads to a

[2] We also searched the smallest penalty parameter for which 4 zero blocks occurred. Then the nonzero loadings were a bit higher, but still considerably smaller than those from Blockwise Simplimax.

local minimum. Therefore, it is recommended to use many differently started runs, and to select the best from these, assuming that this will yield the global minimum.

As with Blockwise Simplimax, it is advised to obtain the optimal solution for a large number of different values of p and to choose among these upon comparing the associated values of f_{BZCCA} for the different values of p. Analogously, one is then recommended to choose that value of p *after* which the value of f_{BZCCA} increases rapidly, as can be seen from a plot of f_{BZCCA} against p, or can be done by the CHull procedure.

The method is computationally quite a bit more time consuming than Blockwise Simplimax. Conjecturing that the matrix **W** obtained from Blockwise Simplimax gives a good indication of which blocks of loadings should be large versus small, we propose the following *alternative* method: Given **W**, minimize (5) subject to the constraint that the blocks $\mathbf{p}_{kr}$ corresponding to the 0 values in the now fixed matrix **W** are 0, and subject to the constraint $\mathbf{T}^{\mathrm{T}}\mathbf{T}=\mathbf{I}$. This method could thus be called "PCA followed by Blockwise Simplimax, followed by Fixed Zero Blocks Constrained Component Analysis", but is denoted here as "Blockwise Zero Constrained CA with fixed **W**". The method is a constrained version of Blockwise Zero Constrained CA, because **W** is being fixed. **W** could be fixed to any desirable structure, but in the present paper we only use the structure resulting from the Blockwise Simplimax solution. The rationale for this constrained approach is dual: On the one hand, it is much more efficient to use a fixed **W**, because the optimization over **W** is the main cause of the large number of local optima typically encountered, and if the **W** used is good, one has a good solution in much smaller computation time. On the other hand, while we realize that this approach may very well give a suboptimal data fit compared to that of Blockwise Zero Constrained CA (and never a better one), it is still possible that the solution obtained is better in the sense of representing the underlying structure in the data. Whether indeed these solutions are good will be tested by simulation studies, as is the subject of the next section.

The two variants of Blockwise Zero Constrained CA were also applied to the example data set from Section 2.2. Inspecting loss function values for the solutions for p=1,…,7 clearly indicated taking p=4. For this, the two methods led to the same solution The results are given in the last two columns of Table 1. Clearly, the nonzero loadings are very similar to those by Blockwise Simplimax, while the zeros are at the same locations as the blocks of small values in Blockwise Simplimax.

3 Simulation studies

In this section we present the results of two simulation studies to compare the performance of PCA followed by Blockwise Simplimax (here briefly denoted as Blockwise Simplimax) and the two Blockwise Zero Constrained CA methods. For comparative purposes, the first one has the very same design as that by Timmerman et al. (2016). They evaluated Blockwise Simplimax and Sparse Group CA, implying that we can – indirectly – make a comparison with Sparse Group CA. The second simulation study employs the same data construction method, but with fewer conditions, and data sizes of 30 × 20 × 8, which are closer to data sizes that are encountered in sensory profiling. Just like Timmerman et al. did, for both designs we also evaluated the performance of the CHull procedures for determining *p,* for all three methods.

3.1 Simulation study 1 (Equal design as Timmerman et al.)

3.1.1 Purpose and Design

The purpose of this study was to compare the new Blockwise Zero Constrained CA methods to Blockwise Simplimax. As we used the very same design, we refer for full detail to Timmerman et al. (2016); here we describe the essence of the design, so as to understand the results in the next subsections.

Performance criteria. The main performance criteria to be studied here are the recovery of the loadings, the recovery of the block structure, and the recovery of the underlying number of zero blocks *p* by CHull. In addition, we inspected the numbers of local optima. To assess the recovery, we used the same measures as Timmerman et al. (2016) did. That is, for the loadings, we used the mean congruence coefficients between the underlying and estimated components (optimized over permutation and reflection). For the recovery of the block structure, we computed the related binary block indicator matrix, and compared that to the underlying block structure indicator, by simply computing the proportion of agreement, that is, the proportion of blocks that had the same binary value. Both recovery measures have values in the range between 0 and 1. In practice, for the congruence a value below .85 can be considered as rather poor (cf Lorenzo-Seva & Ten Berge, 2006). For the proportion of agreement used to assess recovery of the block structure, it should be noted that already by chance alone values in the order of .50 are to be expected. Upon visual inspection of many solutions, it became clear that agreement proportions should also be above .80 for a structure to have a reasonable resemblance to the underlying one.

Design Data were constructed for 100 or 500 cases (sample size), with scores on 4 blocks of variables, consisting of either 6 variables each (equal block size), or 3, 3, 6 and 6 variables, respectively (unequal block size). Noise was included in the data, such that the expected noise percentage per variable was either 25% or 50% (noise level). Finally, the loading matrices were constructed from zeros and large values (sampled randomly from the uniform distribution on [.25, .75]). The location of blocks of zero and large loadings was indicated by the binary block structure matrix, and was varied using four conditions, denoted as, respectively, Easy, Moderate, Difficult and Very Difficult, as follows: $\mathbf{W}^{\text{true}}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$, $\mathbf{W}^{\text{true}}=\begin{bmatrix}1&0&0&0\\1&1&0&0\\0&0&1&0\\0&0&1&1\end{bmatrix}$, $\mathbf{W}^{\text{true}}=\begin{bmatrix}1&1&0&0\\1&0&1&0\\1&0&0&1\\1&0&0&1\end{bmatrix}$, and $\mathbf{W}^{\text{true}}=\begin{bmatrix}1&1&s&s\\1&s&1&s\\1&s&s&1\\1&s&s&1\end{bmatrix}$, with 2/3 of the variables in blocks denoted with an *s* pertaining to zero-loadings, and 1/3 pertaining to relatively small but nonzero loadings. For each cell of the 4 (Block structure) × 2 (Block size) × 2 (Noise level) × 2 (Sample size) factorial design, 100 data matrices **X** were generated according to equation (1), based on random normally distributed component score and error matrices, yielding 3,200 data matrices[3]. As Timmerman et al. did, we centered each simulated data matrix, and subsequently applied PCA followed by Blockwise Simplimax, and the two Blockwise Zero Constrained CA methods.

Methods. Each simulated data set was analyzed by all three methods with *p* equal to the true number of underlying zero (or small) blocks, and for all values of *p* needed to carry out the CHull implementation described by Timmerman et al. (2016). For the Blockwise Simplimax algorithm, the best out of 101 starts (100 random starts and one rational start, based on normalized varimax) was considered. For Blockwise Zero Constrained CA, 102 starts were used (100 random starts, one normalized varimax based start, and one start based on the Blockwise Simplimax solution). Finally, Blockwise Zero Constrained CA with fixed **W** was carried out by first computing the Blockwise Simplimax solution using 101 runs (as above) and then fixing the ensuing **W** and computing the optimal component scores and loadings, using the Blockwise Simplimax loadings as start.

[3] Using the same seeds in the random generators, we hoped to obtain the very same results as Timmerman et al., but this was only partly the case, due to Matlab version differences (between Matlab R2015 and R2016). The results for PCA followed by Blockwise Simplimax therefore differ very slightly from those by Timmerman et al.

3.1.2 Results

3.1.2.1 Sensitivity to local optima

In this section, we investigate the sensitivity of Blockwise Simplimax and Blockwise Zero Constrained CA to local minima. As a proxy for the global minimum we used the best solution out of the 101 or 102 runs of the algorithms, and in both cases did one additional run, started with the true weight structure. If this resulted in a better loss function, we used the latter as proxy[4]. A solution was defined as locally optimal when it exceeded the proxy by more than 10^{-6} times the value of this proxy. For Blockwise Simplimax, using random starts, local optima were found frequently (percentages ranging from 5 to 87 across the cells of the design). Yet, in none of the 3200 replications, all 100 runs led to a local optimum (i.e., at least one out of the 100 values was as low as the lowest value encountered or did not exceed it by more than .0001%). The rational Varimax based start led to a local optimum in 0%, 5%, 77% and 77% of the cases for the four respective block structure conditions. For Blockwise Zero Constrained CA, using random starts, local optima were also found frequently (percentages ranging from 89% to 95% across the cells of the design), but in only 46 out of 3200 replications, all 100 runs led to a local optimum. The rational starts performed considerably better. The Varimax based start led to a local optimum in 0%, 9%, 69% and 84% of the cases for the four respective block structure conditions, while the Blockwise Simplimax started runs led to local optima in respectively 0%, 5%, 12% and 25% of the cases only. All in all, Blockwise Zero Constrained CA is quite sensitive to local optima, but taking the 100 random starts and the two rational starts seems a good strategy to deal with this problem.

3.1.2.2 Recovery of the loadings

The main criterion of interest is the recovery of the loadings, as the loadings dominate the interpretation of solutions. For all cells in the design, the average recovery values for Blockwise Simplimax (PCA+BS in the Figure), Blockwise Zero Constrained CA (BZCCA), and Blockwise Zero Constrained CA with fixed **W** (PCA+BS+FWCCA) are reported in Figure 1. In addition, the results from the Timmerman et al. simulation study for Sparse

[4] Just like Timmerman et al. (2016) did for Blockwise Simplimax, we tested the behavior of the Blockwise Zero Constrained CA methods in noisefree cases. We did so for the same conditions as in the full study, except for leaving out the imperfect block structure "Very Difficult". Furthermore, the component score matrix now was made columnwise orthonormal, so as to allow for a perfect fit, and the loading matrix was rotated by a random rotation matrix. In all cases, the loading matrix and the block structure were recovered perfectly. The random starts led to local optima in 31%, 89% and 83% of the cases in respectively the Easy, Moderate and Difficult conditions, compared to 7%, 86% and 86% for Blockwise Simplimax.

Group CA (SGCA) have been displayed in the same plots. When interpreting the plots it is useful to know that the standard deviations of the recovery values for the different cells ranged between .00 and .12, so standard errors for the averages in Figure 1 range between .00 and .01. Since these are so small, we decided not to clutter up the plot with error bars. Figure 1 reveals clearly that the two new Blockwise Zero Constrained CA methods hardly improve upon Blockwise Simplimax. In fact, in the Very Difficult condition, Blockwise Simplimax actually performs better than the Blockwise Zero Constrained CA methods. This can be attributed to the fact that there were no underlying blocks of all zero loadings, whereas the Blockwise Zero Constrained CA methods do aim at all zero loading blocks. In the other conditions, Blockwise Simplimax is outperformed by one or both of the Blockwise Zero Constrained CA methods, but differences range from .0 to .02, which seems very small for all practical purposes. As already reported by Timmerman et al., Sparse Group CA performed considerably worse than Blockwise Simplimax, and hence also than the Blockwise Zero Constrained CA methods.

3.1.2.3 Recovery of the Block Structure

The average recovery of the Block Structure is displayed in Figure 2. Again, standard errors were relatively small, that is up to .01. Figure 2 gives roughly the same picture as the recovery of the loadings. That is, in terms of recovery of the block structure, improvements by the Blockwise Zero Constrained CA methods over Blockwise Simplimax seem negligible. Furthermore, in the high noise level conditions with the Difficult and Very Difficult block structures, Blockwise Zero Constrained CA performed more poorly than Blockwise Simplimax and Blockwise Zero Constrained CA with fixed **W**. For the Very Difficult condition, this can be understood from the fact that here in the underlying small blocks some loadings are not zero. Compared to Sparse Group CA, the Blockwise Zero Constrained CA methods recovered the block structure better in all cells of the design, except in the Easy conditions, just as Blockwise Simplimax did.

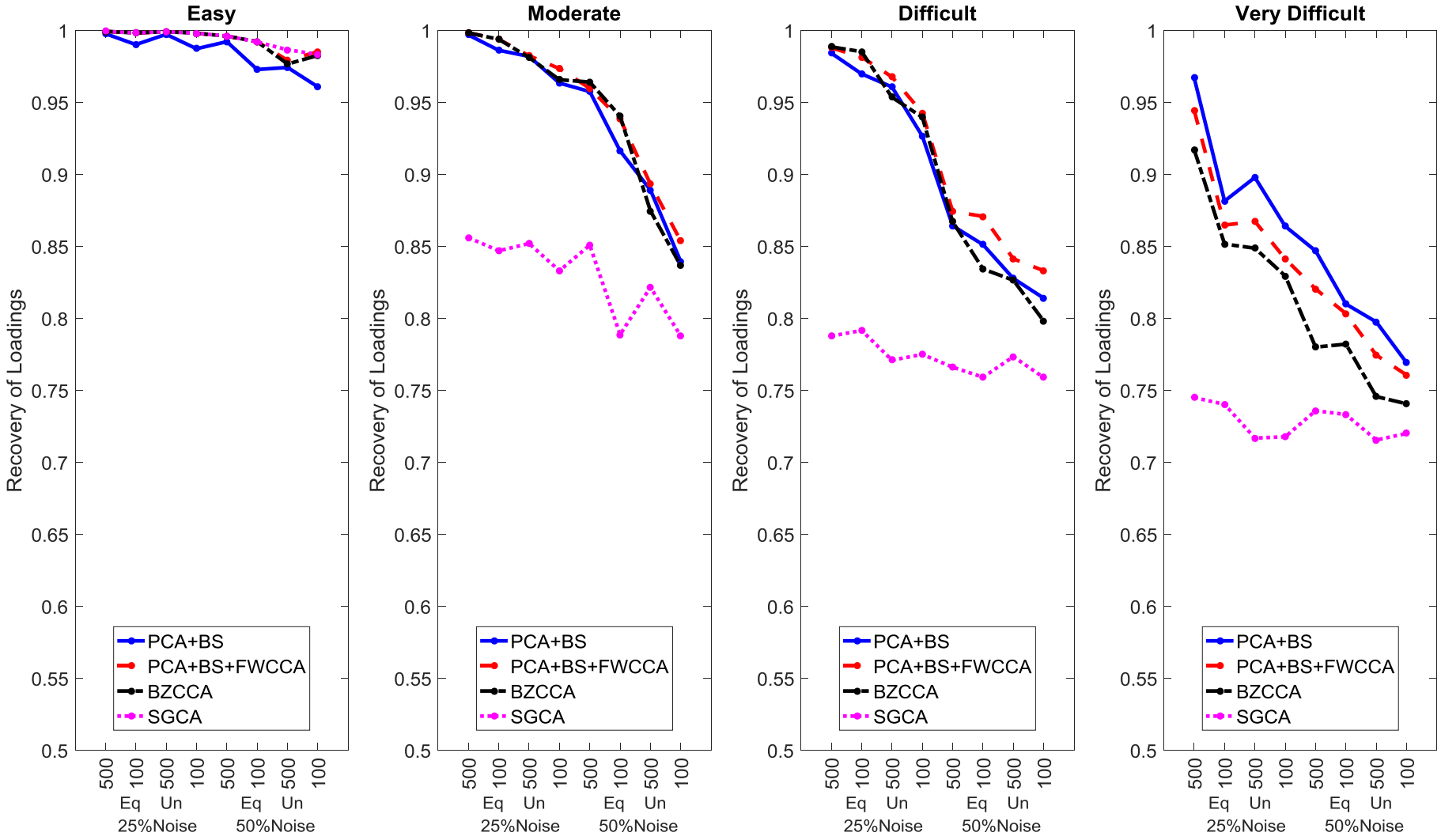

Figure 1. Average recovery of loadings in all cells of design (The four subplots refer to the different block structure conditions, the labels 500 and 100 to the sample size, Eq and Un to the block size conditions, and 25% and 50% noise to the noise conditions. BS is Blockwise Simplimax, FWCCA is Blockwise Zero Constrained CA with fixed W, BZCCA is Blockwise Zero ConstrainedCA, SGCA is Sparse Group CA.)

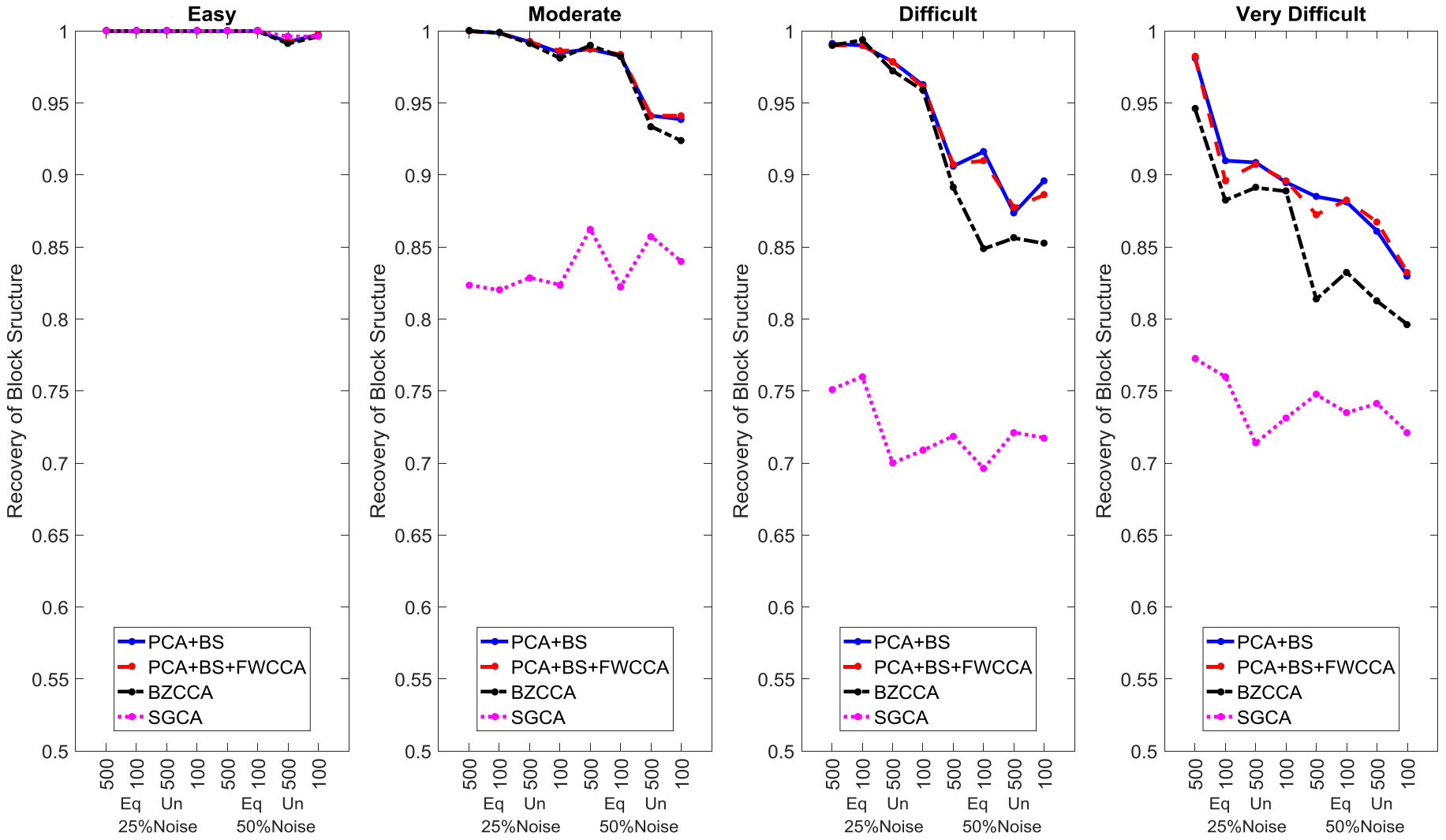

Figure 2. Average recovery of the block structure in all cells of design (The four subplots refer to the different block structure conditions, the labels 500 and 100 to the sample size, Eq and Un to the block size conditions, and 25% and 50% noise to the noise conditions. BS is Blockwise Simplimax, FWCCA is Blockwise Zero Constrained CA with fixed W, BZCCA is Blockwise Zero Constrained CA, SGCA is Sparse Group CA.)

3.1.2.4 Performance of the CHull procedure for selecting the number of small/zero blocks

To investigate the performance of the CHull procedure in selecting the number p of small or 0 blocks (in respectively, Blockwise Simplimax and the two Blockwise Zero Constrained CA methods), we applied the CHull procedure (as described by Timmerman et al. 2016) to the series of outcomes from all three methods, using the "relevant" range[5] of different values of p. For each data set, we recorded whether CHull indicated the true underlying value as the best choice.

In Table 2, the percentages of data sets for which CHull indicated the true value of p as the best solution, are given per cell of the design. It should be noted that now standard errors, which can be computed for each cell separately under a binomial distribution, can be quite large. For instance, for a 50% outcome, the standard error is $\sqrt{(.25/100)}\times 100\%=5\%$, while for outcomes of 10% and 90% they are 3% (using that the standard error for a proportion p based on n replications is $\sqrt{(p(1-p)/n)}$. From Table 2 it can be seen that for all methods CHull was able to recover the underlying p well in all Easy conditions, and in most of the Moderate conditions, but in the Difficult and Very Difficult conditions, recovery of p was very volatile and often poor. Ignoring the cases where CHull led to poor recovery for all methods, the CHull for the Blockwise Zero Constrained CA methods almost always performed comparably to Blockwise Simplimax.

Table 2. Percentages of recovery by CHull of underlying value for p, for the three different methods. (The labels 500 and 100 refer to the sample size, Eq and Un to the block size conditions, and 25% and 50% to the noise conditions.)

	Blockwise Simplimax				**Blockwise Zero Constrained CA**				**Blockwise Zero Constrained CA with fixed W**			
Condition	Easy	Mod	Dif	VDif	Easy	Mod	Dif	VDif	Easy	Mod	Dif	VDif
25%,Eq,500	100	100	96	89	100	100	94	78	100	100	95	81
25%,Eq,100	100	99	93	55	100	99	91	55	100	99	91	56
25%,Un,500	100	97	90	66	100	99	83	39	100	99	83	48
25%,Un,100	100	88	84	54	100	88	74	34	100	89	72	42
50%,Eq,500	100	93	46	37	100	100	27	15	100	100	30	20
50%,Eq,100	100	69	33	25	100	93	19	10	100	90	16	13
50%,Un,500	94	74	42	20	100	95	27	15	100	95	31	11
50%,Un,100	98	47	25	17	100	68	18	16	99	64	17	14

[5] Relevant here means that the analyses were started with values for p equal to the population value, and moved downwards till the fit exceeded 99.01% for Blockwise Simplimax, or 99% of the PCA fit for Blockwise Zero Constrained CA solutions (and the first one to do so was still included in the CHull analysis), and it was moved upwards until a solution with a zero column in the matrix **W** appeared (and also this solution was still included).

Timmerman et al. (2016) suggested that, as in practice one does not know the true number *p*, it is worthwhile to verify the recovery of the loadings even when the incorrect value of *p* has been used. They found that, for Blockwise Simplimax, the recovery using the CHull indicated value of *p* was very close to that by using the underlying *p* consistently. Here we studied this result for Blockwise Simplimax again, as well as for the two Blockwise Zero Constrained CA methods. In Figure 3 we report the resulting average recoveries for all cells of the design, together with the recoveries when using the underlying value of *p* (i.e., as were given in Figure 1). It can be seen that for all three methods, the recovery using the CHull selected solution was not much poorer than when using the (in practice unknown) underlying value of *p.* Differences were smallest for Blockwise Simplimax; the Blockwise Zero Constrained CA methods are somewhat more sensitive to the choices of *p.* It can be concluded that, even though CHull did not recover the values of *p* very well, the CHull based values of *p* were good enough to give good recoveries of the loadings, especially for Blockwise Simplimax.

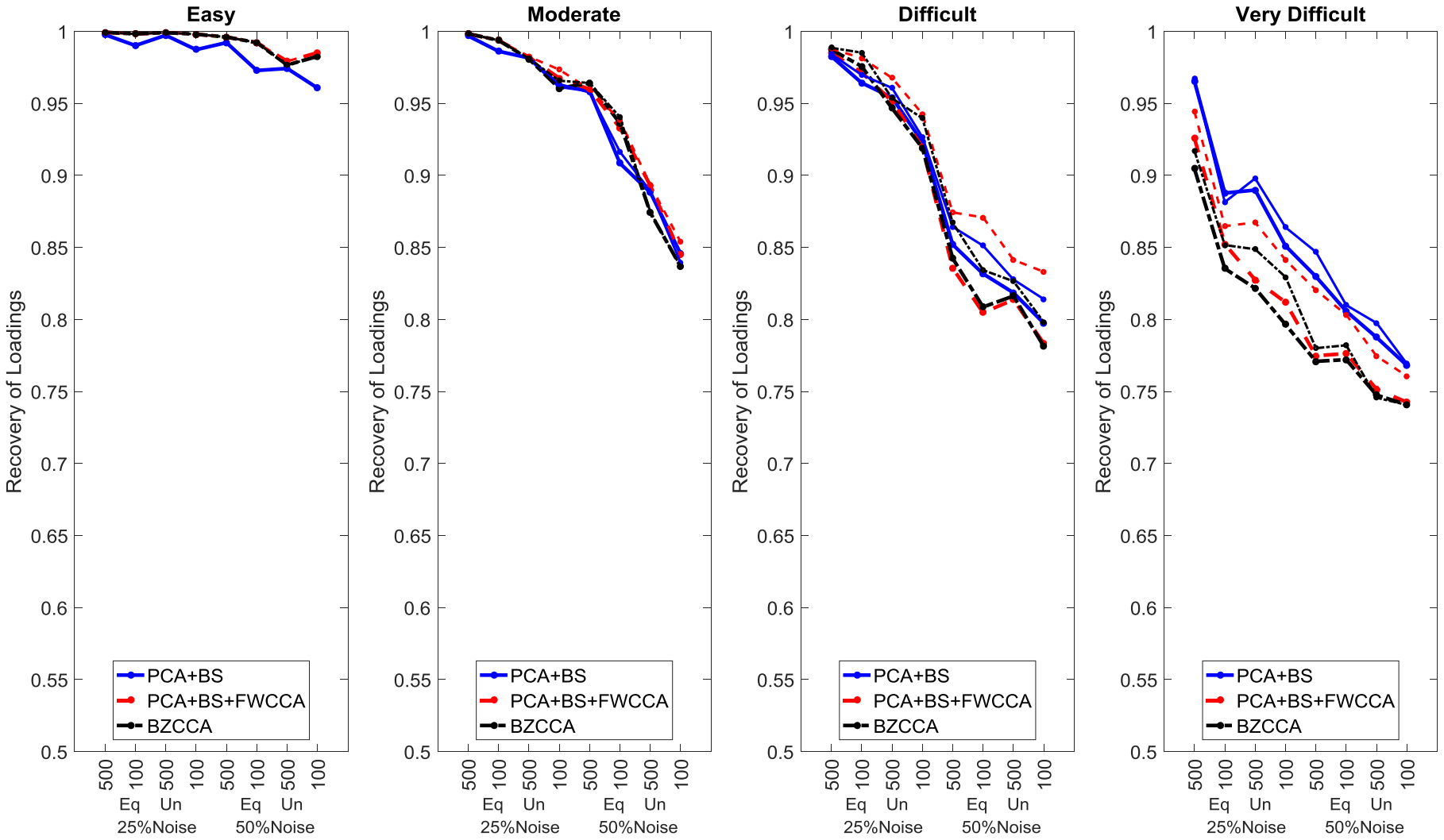

Figure 3. Average recovery of loadings in all cells of design, using both the underlying value of *p* as the CHull selected value of *p*. For each method, the lines referring to the underlying value of *p* are the thinner ones. (The four subplots refer to the different block structure conditions, the labels 500 and 100 to the sample size, Eq and Un to the block size conditions, and 25% and 50% noise to the noise conditions. BS is Blockwise Simplimax, FWCCA is Blockwise Zero CCA with fixed W, BZCCA is Blockwise Zero CCA.)

3.2 Simulation study 2 (data sizes 30×20×8, reminiscent of sensory data)

3.2.1 Purpose and design

The purpose of this second simulation study is to test the methods on data with sizes more typical for sensory profiling analysis. Considering the cheese data by Frøst (2002) as a fairly typical size (30 cheeses, 23 attributes, 8 panelists), we decided to construct data with sample sizes of 30, and having 160 variables, ordered as 20 blocks (pertaining to attributes) of 8 variables each (panelists).

In addition to the size of the data, we also fixed the noise level, to the highest level used in Simulation study 1, i.e. 50%, which we deemed quite realistic for actual practice. Thus we ended up with only one factor in the design, that is, the structure underlying the loadings. We chose three different conditions. In the first condition, denoted as R=2, we used two components, while for each Component we chose 10 zero blocks of loadings, so p=20. In the second condition, denoted as R=3, we had p=36 blocks of zero loadings. The location of the blocks of zero loadings in the two conditions is given in Figure 4.

R=2	*R=3*
1 0	1 0 0
1 0	1 0 0
1 0	1 0 0
1 0	1 0 0
1 0	1 0 0
1 0	1 0 1
1 0	1 0 1
0 1	0 1 0
0 1	0 1 0
0 1	0 1 0
0 1	0 1 0
0 1	0 1 0
0 1	0 1 1
0 1	0 1 1
0 0	0 0 1
0 0	0 0 1
0 0	0 0 1
1 1	0 0 1
1 1	0 0 1
1 1	0 0 1

Figure 4. Location of blocks of zeros (indicated by 0) and nonzero blocks (by 1) in the two conditions.

The *R*=2 and *R*=3 conditions were meant to give fairly realistic underlying structures, which fully adhere to the Blockwise Zero Constrained CA models. They missed one feature that may occur in practice, which is that some panelists may behave idiosyncratically in the use of a particular attribute. Specifically, one panelist may use an attribute as an indicator of an underlying dimension, while all the others don't. Such idiosyncratic behavior implies a sizeable loading for a single panelist within a block of otherwise zero loadings. In the third condition, we constructed data in the same way as in the condition *R*=2, but added four sizeable idiosyncratic loadings *within* zero blocks of loadings (two for the first component, two for the second component, two in cases where a block had zero loadings only for one component, and two in cases where a block had zero loadings for both components). This condition was called "*R*=2_Idio" for short.

As in Timmerman et al. (2016), nonzero loadings were drawn randomly from the uniform [.25,.75] distribution. The four idiosyncratic loadings were taken equal to .50. As in Timmerman et al., the 50% noise level was reached by, for each variable, setting the sum of squares of the noise equal to that of the structural part (i.e., sum of squares of the constructed loadings). For some variables in the present simulation study, the sum of squares of the structural part was 0, so the noise would become 0 as well. To avoid this, in the present simulation study we set the noise sum of squares to .25 in such cases, corresponding in fact to what one would get in case of one nonzero loading of .5.

For the three conditions, 100 data sets were constructed. To these 300 data sets, the three methods were applied, again using all relevant values of *p.*

3.2.2 Results

We first studied the sensitivity to local optima for Blockwise Simplimax, and Blockwise Zero Constrained CA. It turned out that for these data, Blockwise Simplimax was not at all sensitive to local optima: Using random starts, it landed in a local optimum in on average only 1.6%, 3.0% and 1.7% of the runs for the conditions *R*=2, *R*=2_Idio and *R*=3, respectively, and never when using the rational start[6]. Blockwise Zero Constrained CA landed in a local optimum a bit more frequently: on average 2.3%, 4.8% and 9.4% of the randomly started runs for the respective conditions, and never when using either of the two rational starts.

[6] As in Simulation 1, a solution was defined as locally optimal when it exceeded the proxy by more than 10^{-6} times the value of the proxy, which itself was computed in the same way as in Simulation study 1.

The main interest is again in the recovery of the loadings. We first computed these for the methods using the true underlying value of *p.* For Blockwise Simplimax these were on average .96, .96 and .94, for conditions *R*=2, *R*=2_Idio and *R*=3, respectively. For both Blockwise Zero Constrained CA methods they were .98, .96, and .98, respectively. The recovery of the block structure was perfect for all methods, in all conditions. So a first conclusion, on the basis of the overall loadings recovery measures, is that the Blockwise Zero Constrained CA methods performed a little bit better than the already very well performing Blockwise Simplimax. The standard errors for all measures, for all conditions were small (i.e., maximally .001).

For the *R*=2_Idio condition, we also verified how well Blockwise Simplimax actually "recognized" the idiosyncratic loadings[7]. For this purpose, for each analysis we defined a cut-off value above which the loadings were to be considered high. This was simply chosen equal to the mean absolute value of the loadings, as motivated as follows. Considering that in the underlying data 164 nonzero and 156 zero loadings were constructed, one would expect to find roughly half of the loadings to be high and half of them to be low, so the average of all loadings (in absolute sense) should be a reasonable threshold distinguishing large from small. For each data set it was counted how many of the four supposedly idiosyncratic loadings exceeded the cut-off value. This was the case for 381 out of the 400 idiosyncratic loadings constructed. In fact, for 84 (of the 100) data sets, all four idiosyncratic loadings were recognized.

The next question was whether the underlying value of *p* was correctly identified by the CHull procedure. This turned out to be the case for Blockwise Simplimax in 98%, 85% and 98% of the cases in the three respective conditions (*R*=2, *R*=2_Idio and *R*=3). The two Blockwise Zero Constrained CA methods performed almost as well, both leading to respectively, 98%, 85% and 97% perfect recoveries in the three respective conditions. Not surprisingly then, the recovery of the loadings when using the CHull indicated value of *p* was hardly worse than that for analyses using the true underlying value of *p.*

Finally, for these data it was studied to what extent results using any sensible value of *p* actually differed. For this purpose, the recovery of the loadings was computed for the solutions from both Blockwise Simplimax and Blockwise Zero Constrained CA using all relevant (see Footnote 5) values of *p*, and these are given in Figure 5. Taking

[7] Note that the blockwise zero constrained methods by definition will not recognize them, because they aim at setting the full block of loadings to 0.

into account that the underlying values of *p* are 20 for the *R*=2 conditions and 36 for the *R*=3 condition, it is clear that only very rarely the underlying loading structure was not recovered well when *p* was smaller than the true underlying value. When it exceeded that value, this happened far more often, and the recovery deteriorated rapidly as *p* became larger. For Blockwise Simplimax this happened when *p* was quite a bit larger, e.g., more than 6 larger, for Blockwise Zero Constrained CA this already happened when *p* was more than 1 larger.

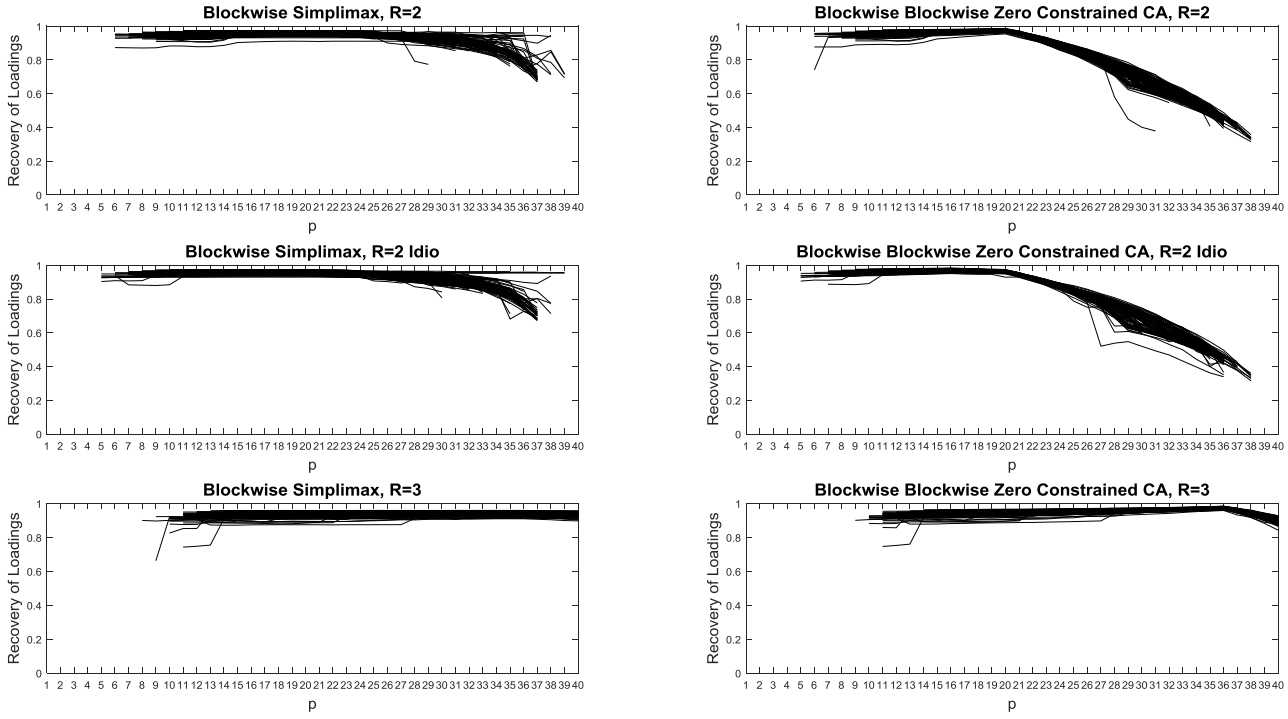

Figure 5. Recovery of loadings by Blockwise Simplimax and Blockwise Zero Constrained CA, for all conditions, and for the complete range of values of *p* used. Note that the underlying *p* equals 20 for *R*=2, *R*=2_Idio, and 36 for *R*=3.

3.3 Discussion of results of the simulation studies

From the simulation results presented above it appears that the recoveries by Blockwise Simplimax and Blockwise Zero Constrained CA do not differ much. In the various conditions in Simulation study 1, they performed either equally well, or Blockwise Simplimax and Blockwise Zero Constrained CA with fixed **W** performed slightly better than Blockwise Zero Constrained CA; in Simulation study 2, the two Blockwise Zero Constrained CA methods performed slightly better than Blockwise Simplimax. The

differences in recovery seem hardly worthwhile. On the other hand, Blockwise Simplimax has the added possibility to identify idiosyncratic loadings. In Simulation study 2 it was seen that it did a very good job in identifying the there constructed idiosyncratic loadings. Furthermore, like in Timmerman et al. (2016), it was seen that the choice of *p* does not seem to be critical for the recovery of the loadings. Using the true underlying value and / or the CHull indicated value gave very similar results, and in the second simulation study it was even seen that a broad range of values for *p* gave virtually the same recovery, as long as *p* did not become (much) too high.

4 Illustrative application of Blockwise Simplimax and the Blockwise Zero Constrained CA methods

To illustrate Blockwise Simplimax and the Blockwise Zero Constrained CA methods, we use the same data as Timmerman et al. used, that is, data from the sensory profiling study on cream cheese (Frøst 2002) (downloaded from http://www.models.life.ku.dk/datasets). These data pertain to 30 cheese samples rated by 8 panelists on 23 attributes. The 30 samples actually consisted of three replications of ten cheeses each (two of which were equal, but this is ignored here). For further details on the study see Bro et al. (2008) and Frøst (2002).

Compared to Timmerman et al. (2016), we give some additional interpretative tools. Specifically, we offer a simple diagnostic for validating the choice of *p*, and we offer a procedure for, in the present case, verifying the reliability of the outcomes. However, to give a complete account of the analysis, we first repeat the results of the analysis by Timmerman et al.

The three-way data set under study here, was treated as a 30 (products) × 184 (panelist-attribute combinations) data set, with 23 blocks with scores of each the 8 panelists on the attribute at hand. We applied PCA followed by Blockwise Simplimax and the two Blockwise Zero Constrained CA methods to the data. These two-way approaches give less concise results than a three-way approach, but offer insight into possible idiosyncratic behavior of panelists in the sense that panelists may treat certain variables differently from how the others treat it. Specifically, Blockwise Simplimax may show that for some panelists attributes usually not related to a particular concept *are* related

to that concept. This then will appear from sizeable loadings in blocks of otherwise small loadings.

Prior to analysis, the scores were centered across the samples and scaled such that, per attribute block, the total sum of squares was 1. Next, we applied PCA with two components, as was suggested by CHull (see Timmerman et al., 2016), and would also be chosen upon inspecting by eye the amounts of variance explained by the R=1,…,8 dimensional solutions: 23.6%, 38.0%, 45.4%, 51.2%, 56.1%, 60.6%, 64.5%, and 68.3%. To the loadings from the two component solution, we applied Blockwise Simplimax with all relevant values of p. The CHull procedure indicated choosing p=32.

To compare solutions for subsequent values of p, for each solution we computed what we term the "fissure". Considering that the method minimizes the p smallest values of $||\mathbf{p}_{kr}||^2$, a rotated solution is easy to interpret if these p smallest values indeed are relatively small compared to the other $JR-p$ values $||\mathbf{p}_{kr}||^2$. Indeed, it would be desirable that the p smallest values are all considerably smaller than the $JR-p$ largest values. This is the case when the maximum of the smallest values is considerably smaller than the minimum of the highest values. So to assess this difference, we compute the "fissure", defined as the minimum of the JR-p largest $||\mathbf{p}_{kr}||^2$'s minus the maximum of the p smallest $||\mathbf{p}_{kr}||^2$'s. In the present case, the fissure for the different values of p is as plotted in Figure 6. In Figure 6, also the largest sum of squares of the small blocks is plotted. It can be seen that, with increasing p the latter increases. The fissure, however, is relatively small for many values of p, but has two clearly high values: These were found for p=32 and p=38. Thus, for the solutions related to these values of p, one could say that the p small blocks are really considerably smaller than the $46-p$ large blocks. For the solution for p=32, the largest small value of $||\mathbf{p}_{kr}||^2$ was .19 while the smallest large value was .28. For p=38, the largest small value was .30 while the smallest large value was .39. To see this in perspective, we compared them to the average value of the $||\mathbf{p}_{kr}||^2$'s which equals .19 (Note that this holds for every rotation, because rotations do not affect the total sum of $||\mathbf{p}_{kr}||^2$'s). Hence in the p=32 solution, the largest small block is equal to the average $||\mathbf{p}_{kr}||^2$-value, while for the p=38 solution it highly exceeds that average value, and can therefore hardly be considered small. We can conclude that only p=32 leads to a relatively big fissure, and to a proper distinction between blocks of small and large loadings.

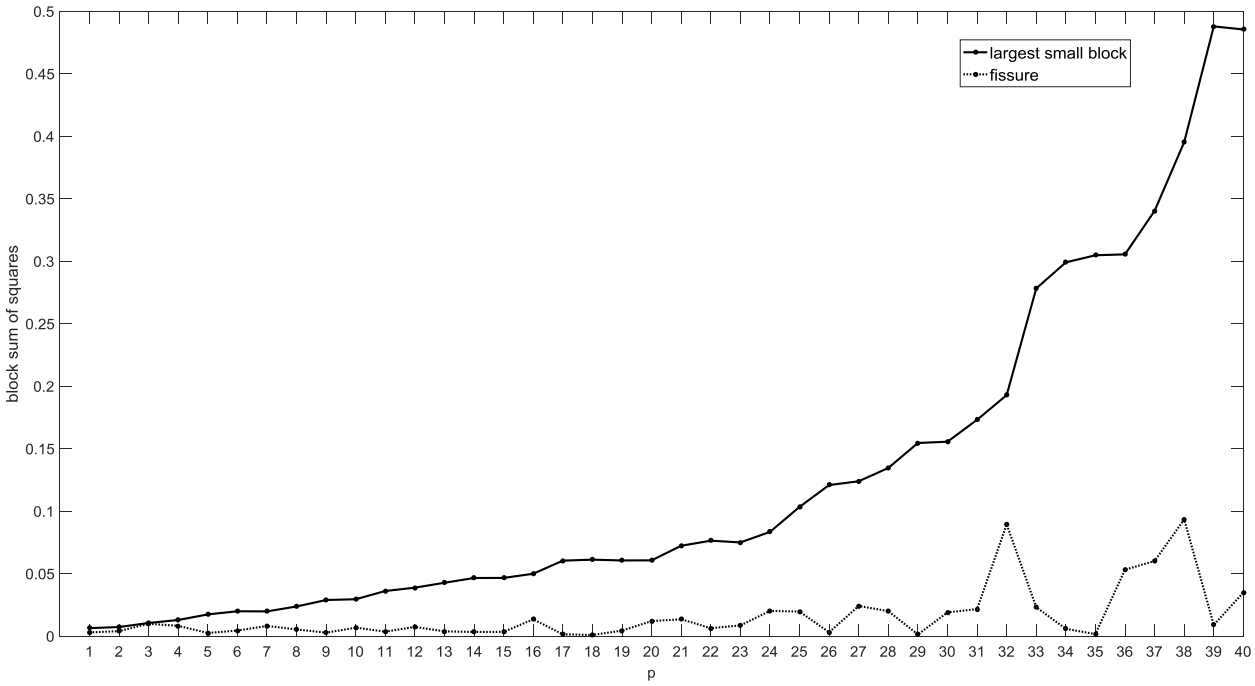

Figure 6. The largest value of the *p* smallest blockwise sums of squared loadings (solid line), and the fissure (dashed line)

The Blockwise Simplimax rotated loadings on the two components are reproduced in Table 3, where the *p*=32 blocks of small loadings are made less visible by shading their cells. For structured plots of loadings, where loadings from different panelist for the same attribute are linked to each other by means of star plots, see Timmerman et al. (2016, Figure 2). To interpret the components, we focus on the non-shaded blocks, which are the ones with the highest sums of squared loadings. Component 1 is thus interpreted as Firm (H-Resistance, M-Firm and M-Resistance) versus Soft/Shiny (E-Shiny and M-Melt). Component 2 is interpreted as Creamy (M-Butter, E-Yellow, M-Fat, M-Salt, M-Creaminess and M-Cream) versus Sour/Chalky (E-White, M-Sour and M-Chalky).

Table 3. Loadings after Blockwise SIMPLIMAX rotation of the SCA solution with 2 components and *p*=32 zero-vectors. To ease the presentation, the loadings per attribute of the 8 panelists are positioned next to each other. Loadings that are rotated to a small value (i.e., associated with a value of 0 in W) are printed in a shaded (grey) cell; loadings larger than .25 in absolute value are printed in bold face.

	Component 1								Component 2							
Panelist	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
N-Cream	.02	-.07	.05	.14	-.02	.04	-.00	-.08	.08	.24	.05	.21	-.03	-.03	-.06	.08
N-Acidic	-.03	-.01	.03	.02	-.00	.08	.11	.01	-.03	-.20	-.01	-.02	-.02	.02	-.04	-.03
N-Butter	.03	-.02	.03	.09	.04	-.01	-.04	-.02	.15	.16	.03	.22	.03	.07	-.01	-.01
N-Old milk	-.03	.02	-.04	.05	.01	-.04	-.02	.01	**-.28**	-.14	-.04	-.19	-.01	-.11	-.09	-.01
E-White	-.01	.10	.02	.12	.02	.03	.15	.13	-.20	**-.25**	-.15	-.24	-.17	-.15	-.20	-.10
E-Grey	.04	.04	.05	.03	.03	.03	-.04	.17	.03	-.12	-.09	-.06	-.15	.01	-.14	**-.32**
E-Yellow	-.03	-.11	.01	-.05	.01	-.06	-.05	-.02	**.27**	**.33**	.16	.21	.18	.19	.23	.19
E-Green	.07	-.08	.03	-.07	.10	.00	.04	.04	.09	.01	-.04	**.26**	-.01	-.06	-.07	.00
H-Resistance	**.29**	**.26**	**.27**	**.30**	**.25**	**.31**	**.26**	**.34**	-.08	-.01	-.09	.07	.01	.05	-.03	.17
E-Grainy	.20	.06	.14	**-.25**	.01	.18	.02	.00	-.04	-.09	.03	-.05	-.15	-.17	-.01	-.08
E-Shiny	**-.36**	-.24	-.22	**-.37**	**-.31**	**-.28**	-.18	-.17	.01	.04	.00	-.10	.00	-.02	.00	-.03
M-Firm	.17	.23	**.29**	**.32**	**.29**	**.30**	**.39**	**.35**	-.05	-.07	-.12	-.10	-.05	-.07	-.11	-.00
M-Melt down	-.20	-.09	**-.29**	**-.33**	**-.26**	**-.29**	-.23	-.18	.03	.15	.12	.14	.09	.08	.06	.10
M-Resistance	.11	.07	.21	**.36**	.22	**.29**	**.36**	**.28**	-.05	-.14	-.12	-.15	-.11	-.14	-.03	-.04
M-Creaminess	-.07	.11	.04	**.29**	.08	.05	-.03	.02	**.40**	.01	.05	-.15	.09	-.05	**.30**	.12
M-Grainy	.14	-.09	-.02	-.07	-.04	-.09	-.03	-.03	-.05	.22	.03	.05	.02	.06	.01	.01
M-Chalky	.06	-.01	.10	.02	.03	.11	.02	.07	**-.33**	-.21	-.18	**-.37**	-.21	-.23	**-.28**	-.14
M-Cream	.00	.03	.05	.01	.05	-.01	.06	-.03	.19	.22	.17	**.36**	.05	-.01	.21	.10
M-Fat	-.05	-.03	.04	.11	-.01	.20	.03	.04	**.42**	**.30**	.11	**.27**	.09	-.07	**.35**	.09
M-Butter	.00	-.01	-.00	.11	-.04	-.05	-.02	.00	**.28**	**.25**	**.27**	**.30**	.13	**.34**	**.37**	.08
M-Salt	-.14	-.05	-.04	-.16	-.09	-.22	-.16	.04	**.25**	.20	.04	.14	.01	**.44**	.22	.05
M-Sour	-.01	-.00	-.05	-.13	-.05	-.08	-.06	.05	-.02	**-.35**	**-.26**	-.14	-.15	-.10	-.18	-.16
M-Sweet	-.02	-.01	-.02	.04	.00	-.07	.21	-.01	.16	.14	-.02	.13	.10	-.12	**.32**	.05

We also computed the solutions by the two Blockwise Zero Constrained CA methods for these data. These two gave virtually identical results, therefore we focus here on only one solution of Blockwise Zero Constrained CA (i.e., without fixed **W**). Interestingly, applying the CHull criterion also *p*=32 was indicated. This solution yielded a fit of 29.0%, which is considerably lower than that of PCA. The results are given in Table 4. The most salient difference is that the small loadings resulting from Blockwise Simplimax, now all become 0 (due to the zero constraints); surprisingly, the other loadings were virtually the same across the two solutions: they never differed by more than .03. Thus, in this case, the effect of replacing the Blockwise Simplimax solution by that from Blockwise Zero Constrained CA is only that interpretation becomes simpler, because small loadings actually become zero, but apparently no new insights (which could possibly have been masked in the Blockwise Simplimax solution) are gained by

applying Blockwise Zero Constrained CA. On the other hand, making all blocks of small loadings zero, actually entails a loss of information, because in the blocks of small loadings some idiosyncratic loadings may be found, and hence idiosyncratic panelist behavior may be spotted.

Table 4. Loadings from BZCCA with 2 components and *p*=32 zero-vectors. To ease the presentation, the loadings per attribute of the 8 panelists are positioned next to each other. Blocks of zero loadings are printed in shaded (grey) cells; loadings larger than .25 in absolute value are printed in bold face.

	Component 1								Component 2							
Panelist	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
N-Cream	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N-Acidic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N-Butter	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N-Old milk	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E-White	0	0	0	0	0	0	0	0	-.20	**-.27**	-.16	-.24	-.18	-.16	-.21	-.10
E-Grey	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E-Yellow	0	0	0	0	0	0	0	0	**.28**	**.36**	.16	.21	.19	.20	**.25**	.20
E-Green	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H-Resistance	**.30**	**.27**	**.27**	**.31**	**.27**	**.31**	**.28**	**.34**	0	0	0	0	0	0	0	0
E-Grainy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E-Shiny	**-.37**	-.24	-.23	**-.37**	**-.31**	**-.28**	-.19	-.17	0	0	0	0	0	0	0	0
M-Firm	.17	.23	**.28**	**.33**	**.30**	**.29**	**.38**	**.36**	0	0	0	0	0	0	0	0
M-Melt down	-.20	-.09	**-.29**	**-.33**	**-.27**	**-.29**	-.22	-.19	0	0	0	0	0	0	0	0
M-Resistance	.12	.06	.21	**.35**	.24	**.29**	**.36**	**.28**	0	0	0	0	0	0	0	0
M-Creaminess	0	0	0	0	0	0	0	0	**.40**	.00	.05	-.15	.10	-.06	**.29**	.12
M-Grainy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-Chalky	0	0	0	0	0	0	0	0	**-.33**	-.21	-.18	**-.36**	-.20	-.22	**-.28**	-.14
M-Cream	0	0	0	0	0	0	0	0	.18	.22	.17	**.36**	.05	-.01	.21	.11
M-Fat	0	0	0	0	0	0	0	0	**.42**	**.30**	.11	**.26**	.10	-.08	**.33**	.09
M-Butter	0	0	0	0	0	0	0	0	**.28**	**.27**	**.27**	**.29**	.13	**.34**	**.35**	.08
M-Salt	0	0	0	0	0	0	0	0	**.26**	.20	.05	.16	.03	**.45**	.23	.06
M-Sour	0	0	0	0	0	0	0	0	-.03	**-.36**	**-.26**	-.15	-.16	-.11	-.17	-.16
M-Sweet	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

As can be seen from Table 3, in the Blockwise Simplimax solution, some panelists show large loadings in blocks of small loadings. With .25 as cut-off value, the six cases in point are Panelist 1 for N-Old milk on component 2, Panelist 4 for E-Grainy and M-Creaminess on Component 1, and E-Green on Component 2, Panelist 7 for M-Sweet on Component 2, and Panelist 8 for E-grey on Component 2. So apparently these panelists use the respective attributes differently from the other panelists. However, one may wonder to what extent these differences are reliable. To interpret these loadings, one

might like to have at least some measure of confidence, like confidence intervals, but the current sample of 30 products is by no means a random sample from a population, so the usual statistical procedures cannot be used. Here, however, we have replicates of the sample cheeses, and we used this to get some indication of the reliability of outcomes as follows.

Each of the 30 products was one of three replicates of samples of a particular cheese. Now if results from the overall study are reliable, one would expect them to hold irrespective of which replicate is being used. Indeed, one would expect them to hold for data sets consisting of only one replicate from each product. Therefore, we constructed three data sets obtained after assigning, for each product, the data for the three replicates to different data sets. Thus we obtained three data sets simply by assigning the first replicate of each product to data set 1, the second to data set 2, and the third to data set 3. Next these data sets were independently analyzed by PCA followed by Blockwise Simplimax, and the value of p was obtained by the CHull procedure. This led to three different values of p. Since we had found before that the value of p does not seem critical, we did no attempt to take equal p's, and simply used the solutions obtained with the CHull indicated p's. To make loadings size comparable[8], we scaled the loading matrices such that their sums of squares equaled that for the full data set. Next we permuted and reflected columns of loadings such that they optimally resembled the solution for the full data set (in terms of congruences). Having thus made solutions comparable, we now considered the variation between the three solutions as an indication of the reliability. For each loading we computed the standard deviation across the three data sets (hence across the three associated loadings), and used this as a relative measure of reliability of the loadings. The standard deviations are given in Table 5. We see that quite a few loadings differ considerably across the replications, while others are fairly constant. Although all standard deviations could be inspected, we focus on those for the loadings identified as "idiosyncratic" above, and indicated in Table 5 by boxes around the loadings and standard deviations.

[8] PCA's on smaller samples generally lead to higher loadings.

Table 5. Loadings from Blockwise Simplimax with, in italics, the standard deviations across results for three replicates.

	Component 1								Component 2							
Panelist	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
N-Cream	.02	-.07	.05	.14	-.02	.04	-.00	-.08	.08	.24	.05	.21	-.03	-.03	-.06	.08
	.02	*.04*	*.06*	*.14*	*.02*	*.08*	*.17*	*.08*	*.06*	*.05*	*.04*	*.16*	*.04*	*.08*	*.06*	*.11*
N-Acidic	-.03	-.01	.03	.02	-.00	.08	.11	.01	-.03	-.20	-.01	-.02	-.02	.02	-.04	-.03
	.01	*.04*	*.03*	*.10*	*.09*	*.10*	*.03*	*.05*	*.10*	*.09*	*.01*	*.10*	*.11*	*.13*	*.12*	*.01*
N-Butter	.03	-.02	.03	.09	.04	-.01	-.04	-.02	.15	.16	.03	.22	.03	.07	-.01	-.01
	.04	*.04*	*.03*	*.06*	*.13*	*.05*	*.18*	*.02*	*.08*	*.10*	*.07*	*.07*	*.04*	*.02*	*.11*	*.06*
N-Old milk	-.03	.02	-.04	.05	.01	-.04	-.02	.01	**-.28**	-.14	-.04	-.19	-.01	-.11	-.09	-.01
	.12	*.09*	*.03*	*.02*	*.05*	*.07*	*.04*	*.06*	*.08*	*.14*	*.06*	*.15*	*.06*	*.07*	*.03*	*.01*
E-White	-.01	.10	.02	.12	.02	.03	.15	.13	-.20	**-.25**	-.15	-.24	-.17	-.15	-.20	-.10
	.01	*.05*	*.06*	*.06*	*.01*	*.04*	*.04*	*.03*	*.06*	*.03*	*.05*	*.08*	*.02*	*.03*	*.09*	*.07*
E-Grey	.04	.04	.05	.03	.03	.03	-.04	.17	.03	-.12	-.09	-.06	-.15	.01	-.14	**-.32**
	.04	*.07*	*.04*	*.09*	*.03*	*.06*	*.07*	*.21*	*.04*	*.08*	*.02*	*.12*	*.07*	*.06*	*.14*	*.24*
E-Yellow	-.03	-.11	.01	-.05	.01	-.06	-.05	-.02	**.27**	**.33**	.16	.21	.18	.19	.23	.19
	.03	*.08*	*.01*	*.07*	*.02*	*.04*	*.07*	*.08*	*.04*	*.08*	*.05*	*.02*	*.04*	*.05*	*.04*	*.04*
E-Green	.07	-.08	.03	-.07	.10	.00	.04	.04	.09	.01	-.04	**.26**	-.01	-.06	-.07	.00
	.02	*.06*	*.04*	*.15*	*.24*	*.04*	*.05*	*.11*	*.05*	*.05*	*.07*	*.04*	*.06*	*.06*	*.02*	*.08*
H-Resistance	**.29**	**.26**	**.27**	**.30**	**.25**	**.31**	**.26**	**.34**	-.08	-.01	-.09	.07	.01	.05	-.03	.17
	.03	*.06*	*.03*	*.04*	*.02*	*.03*	*.03*	*.03*	*.06*	*.03*	*.05*	*.04*	*.01*	*.06*	*.02*	*.08*
E-Grainy	.20	.06	.14	**-.25**	.01	.18	.02	.00	-.04	-.09	.03	-.05	-.15	-.17	-.01	-.08
	.04	*.02*	*.05*	*.09*	*.04*	*.11*	*.04*	*.01*	*.02*	*.05*	*.04*	*.17*	*.07*	*.06*	*.07*	*.07*
E-Shiny	**-.36**	-.24	-.22	**-.37**	**-.31**	**-.28**	-.18	-.17	.01	.04	.00	-.10	.00	-.02	.00	-.03
	.04	*.06*	*.05*	*.04*	*.01*	*.03*	*.02*	*.05*	*.09*	*.01*	*.01*	*.03*	*.03*	*.05*	*.05*	*.04*
M-Firm	.17	.23	**.29**	**.32**	**.29**	**.30**	**.39**	**.35**	-.05	-.07	-.12	-.10	-.05	-.07	-.11	-.00
	.04	*.02*	*.02*	*.02*	*.04*	*.01*	*.05*	*.02*	*.02*	*.03*	*.02*	*.02*	*.08*	*.02*	*.06*	*.02*
M-Meltdown	-.20	-.09	**-.29**	**-.33**	**-.26**	**-.29**	-.23	-.18	.03	.15	.12	.14	.09	.08	.06	.10
	.04	*.03*	*.06*	*.06*	*.09*	*.05*	*.12*	*.03*	*.04*	*.07*	*.03*	*.07*	*.11*	*.07*	*.09*	*.05*
M-Resistance	.11	.07	.21	**.36**	.22	**.29**	**.36**	**.28**	-.05	-.14	-.12	-.15	-.11	-.14	-.03	-.04
	.01	*.07*	*.04*	*.04*	*.10*	*.04*	*.09*	*.01*	*.05*	*.05*	*.01*	*.03*	*.08*	*.06*	*.07*	*.03*
M-Creaminess	-.07	.11	.04	**.29**	.08	.05	-.03	.02	**.40**	.01	.05	-.15	.09	-.05	**.30**	.12
	.04	*.03*	*.04*	*.02*	*.06*	*.06*	*.02*	*.03*	*.04*	*.03*	*.03*	*.09*	*.03*	*.07*	*.07*	*.02*
M-Grainy	.14	-.09	-.02	-.07	-.04	-.09	-.03	-.03	-.05	.22	.03	.05	.02	.06	.01	.01
	.03	*.05*	*.02*	*.06*	*.04*	*.02*	*.01*	*.07*	*.06*	*.08*	*.02*	*.01*	*.05*	*.06*	*.03*	*.07*
M-Chalky	.06	-.01	.10	.02	.03	.11	.02	.07	**-.33**	-.21	-.18	**-.37**	-.21	-.23	**-.28**	-.14
	.06	*.08*	*.00*	*.06*	*.04*	*.06*	*.02*	*.02*	*.05*	*.11*	*.04*	*.08*	*.03*	*.01*	*.07*	*.01*
M-Cream	.00	.03	.05	.01	.05	-.01	.06	-.03	.19	.22	.17	**.36**	.05	-.01	.21	.10
	.09	*.06*	*.06*	*.08*	*.03*	*.03*	*.07*	*.06*	*.06*	*.04*	*.02*	*.11*	*.04*	*.16*	*.05*	*.09*
M-Fat	-.05	-.03	.04	.11	-.01	.20	.03	.04	**.42**	**.30**	.11	**.27**	.09	-.07	**.35**	.09
	.11	*.05*	*.06*	*.02*	*.02*	*.02*	*.05*	*.03*	*.05*	*.08*	*.02*	*.04*	*.06*	*.03*	*.01*	*.06*
M-Butter	.00	-.01	-.00	.11	-.04	-.05	-.02	.00	**.28**	**.25**	**.27**	**.30**	.13	**.34**	**.37**	.08
	.05	*.04*	*.02*	*.03*	*.03*	*.05*	*.05*	*.02*	*.03*	*.05*	*.03*	*.05*	*.04*	*.01*	*.04*	*.06*
M-Salt	-.14	-.05	-.04	-.16	-.09	-.22	-.16	.04	**.25**	.20	.04	.14	.01	**.44**	.22	.05
	.02	*.05*	*.02*	*.05*	*.02*	*.17*	*.07*	*.02*	*.02*	*.02*	*.01*	*.03*	*.03*	*.13*	*.06*	*.03*
M-Sour	-.01	-.00	-.05	-.13	-.05	-.08	-.06	.05	-.02	**-.35**	**-.26**	-.14	-.15	-.10	-.18	-.16
	.02	*.07*	*.08*	*.04*	*.03*	*.02*	*.05*	*.03*	*.05*	*.06*	*.01*	*.07*	*.02*	*.05*	*.03*	*.05*
M-Sweet	-.02	-.01	-.02	.04	.00	-.07	.21	-.01	.16	.14	-.02	.13	.10	-.12	**.32**	.05
	.08	*.04*	*.01*	*.07*	*.07*	*.16*	*.10*	*.08*	*.03*	*.05*	*.01*	*.01*	*.04*	*.18*	*.03*	*.07*

Now we see that the idiosyncratic loading for E-grey is very unstable, and cannot be taken too seriously. On the other hand, the idiosyncratic loadings of Panelist 4 for M-

Creaminess on Component 1, and for E-Green on Component 2, and that of Panelist 7 for M-Sweet on Component 2 were very stable, and can be taken seriously. Thus, we can state that Panelist 4 is the only one who associates M-Creaminess with Firmness (the main definer of Component 1), and E-Green with the Creaminess dimension represented by Component 2, while Panelist 7 is the only one who relatively strongly associates Sweetness with this Creaminess dimension. The other two idiosyncratic loadings (of N-Old milk and E-Grainy) appeared somewhat more unstable, so it is hard to say whether these are to be taken seriously or not.

It can be concluded that the solution from PCA followed by Blockwise Simplimax, can easily be interpreted on the basis of the blocks of larger loadings, while idiosyncratic behavior of panelists can be identified by looking for large loadings in small blocks. The latter can to some extent be validated by means of comparison of solutions from different replications. The choice of the number of small blocks could be determined well by means of the CHull procedure, and was validated by means of the newly introduced fissure search approach. Application of the Blockwise Zero Constrained CA methods yielded the same information as Blockwise Simplimax concerning which are the small blocks, but did not yield any further insights above Blockwise Simplimax: The large loadings were virtually equal.

5 Concluding remarks and Discussion

In the present paper, we proposed two new methods for obtaining blockwise small loadings: Blockwise Zero Constrained CA and Blockwise Zero Constrained CA with fixed **W**. By means of two simulations studies, and application to an empirical example, we compared these methods to Blockwise Simplimax. It was found that the Blockwise Zero Constrained CA methods performed slightly better than Blockwise Simplimax as far as recovery of the loadings and the block structure is concerned, but differences do not seem of any practical value. On the other hand, using the blockwise zero constraints, one loses insight into possible idiosyncratic loadings within blocks of small loadings. It has been demonstrated by an empirical example how such interesting information can be obtained with Blockwise Simplimax, and a method for inspecting the reliability of such results has been proposed and demonstrated. Furthermore, in Simulation study 2 it was

found that the great majority of constructed idiosyncratic loadings was identified by the method.

As to the choice of the number of blocks to be made small, it has been seen that, like for Blockwise Simplimax, the recovery by the constrained methods is not strongly dependent on the use of the true number of underlying zero blocks, even though they depend more on it than Blockwise Simplimax does. The recovery of this number of blocks by means of the CHull procedure was quite well in some conditions, but quite poor in others. However, correct recovery of p in itself appeared not to be crucial in order to obtain a good loading matrix. The chosen value of p, in practice, however, will also determine which blocks are considered large or small, and hence are used or discarded in the interpretation. Therefore, it is useful to validate this choice by means of the fissure statistic introduced in Section 4. This assesses the difference between what is denoted as small blocks and as large blocks, and one can thus check whether the chosen solution corresponds to a relatively big fissure. This seems to be a useful additional tool in choosing which blocks are to be considered large or small blocks while interpreting a solution.

Our main conclusion is that, for interpretational purposes, PCA followed by Blockwise Simplimax is to be preferred over the Blockwise Zero Constrained CA approaches: It is virtually as good as the Blockwise Zero Constrained CA methods in identifying a structure of blocks of small loadings, but in addition gives information on idiosyncratic loadings within blocks of small loadings, thus identifying deviant panelist behavior. Only if one's goal is merely to identify the block structure, and there is no interest in deviant panelist behavior, the Blockwise Zero Constrained CA methods could be preferred, especially the variant with fixed **W** (based on Blockwise Simplimax). In such cases of merely identifying the block structure associated with the attributes, one might think that it would be actually unnecessary to analyze the full data set: Averaging across panelists gives a nice products × attribute data matrix, which could be analyzed by PCA and ordinary Simplimax. To see whether this indeed works, this procedure has been tested on the same simulated data, and the empirical data set. It was found that, in the data from the first simulation study, the block structure was recovered more poorly than by the other methods, and in the analysis of the example data, a slight difference in block structure was obtained; in the second simulation study, the same block structure

was found throughout. Therefore, the simple method of averaging across panelists cannot be trusted to be good enough in all cases. Hence, analyzing the full data, using PCA followed by Blockwise Simplimax or Blockwise Zero Constrained CA seems to be preferred over the averaging method anyhow.

The methodology proposed here is meant for "fixed vocabulary" profiling data. For such data we have the special property that all blocks consist of the same number of variables, i.e. all *K* attribute × panelist combinations related to one attribute. The methods as originally proposed are, however, not limited to situations with equal block size, as has even been used in the first simulation study. In case of fixed vocabulary data, this might be useful in cases where data for some attribute × panelist combinations are incomplete, because after deleting the scores for such a variable, one can still do the blockwise analyses. However, it also opens the door to the analysis of Free Choice profiling data (Williams & Langron, 1984). That is, in case (expert) knowledge is available on which attributes proposed by different panelists could be considered more or less equivalent, one could use this in the set-up of the data by reordering the variables such that equivalent attributes are collected in blocks. The total number of blocks thus composed hence depends on the total number of equivalence classes that can be defined sensibly, and these classes will usually have different numbers of variables. The thus ordered data can next be analyzed by all methods described in this paper. If so desired, the blocks can be weighted to account for the influence of block size.

6 References

Adachi, K., & Trendafilov, N. T. (2016). Sparse principal component analysis subject to prespecified cardinality of loadings. *Computational Statistics*, *31*(4), 1403-1427.

Bro, R., Qannari, E. M., Kiers, H. A. L., Næs, T., & Frøst, M. B. (2008). Multi-way models for sensory profiling data. *Journal of Chemometrics*, *22(1*), 36-45.

De Roover, K., Timmerman, M. E., Van Mechelen, I., & Ceulemans, E. (2013). On the added value of multiset methods for three-way data analysis. *Chemometrics and Intelligent Laboratory Systems*, *129*, 98-107.

Dijksterhuis, G. (1995). Assessing panel consonance. *Food Quality and Preference*, *6(1)*, 7-14.

Dijksterhuis, G., & Punter, P. (1990). Interpreting generalized procrustes analysis 'analysis of variance'tables. *Food Quality and Preference*, *2(4)*, 255-265.

Frøst, M. B. (2002). *The influence of fat content on sensory properties and consumer perception of dairy products.* (Unpublished The Royal Veterinary and Agricultural University).

Harshman, R.A. (1970). Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multi-mode factor analysis. University of California at Los Angeles, *UCLA Working Papers in Phonetics, 16,* 1-84.

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, *23(3)*, 187-200.

Kiers, H. A. L. (1994). SIMPLIMAX: Oblique rotation to an optimal target with simple structure. *Psychometrika*, *59(4)*, 567-579.

Krijnen, W.P. & Kiers, H.A.L. (1993) Clustered variables in PARAFAC. In: J.H.L. Oud & R.A.W. van Blokland-Vogelesang (Eds.) *Advances in longitudinal and multivariate analysis in the behavioral sciences: Proceedings of the SMABS 1992 conference* (pp.165-177). Nijmegen; ITS.

Kroonenberg, P. M., & De Leeuw, J. (1980). Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, *45(1)*, 69-97.

Lavit, C. (1988). *Analyse conjointe de tableaux quantitatifs.* Masson, Paris.

Lorenzo-Seva, U., & Ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, *2(2)*, 57-64.

Qannari, E. M., Wakeling, I., & MacFie, H. J. (1995). A hierarchy of models for analysing sensory data. *Food quality and preference*, *6*(4), 309-314.

Timmerman, M. E., Kiers, H. A. L., & Ceulemans, E. (2016). Searching components with simple structure in simultaneous component analysis: Blockwise Simplimax rotation. *Chemometrics and Intelligent Laboratory Systems*, *156*, 260-276.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31(3)*, 279-311.

Van Deun, K., Wilderjans, T. F., van, d. B., Antoniadis, A., & Van Mechelen, I. (2011). A flexible framework for sparse simultaneous component based data integration. *BMC Bioinformatics, 12*(1), 448.

Wilderjans, T. F., & Cariou, V. (2016). CLV3W: A clustering around latent variables approach to detect panel disagreement in three-way conventional sensory profiling data. *Food Quality and Preference*, *47*, 45-53.

Wilderjans, T. F., Ceulemans, E., & Meers, K. (2013). CHull: A generic convex-hull-based model selection method. *Behavior Research Methods, 45*(1), 1-15.

Williams, A. A., & Langron, S. P. (1984). The use of free-choice profiling for the evaluation of commercial ports. *Journal of the Science of Food and Agriculture*, *35*(5), 558-568.

Conflict of Interest: Kiers, Timmerman and Ceulemans declare that they have no conflict of interest.

Compliance with Ethical Requirements: Kiers, Timmerman and Ceulemans declare that they comply with Elsevier's ethical policies.

Funding: The research leading to the results reported in this paper was sponsored in part by a research grant from the Fund for Scientific Research-Flanders (FWO, Project No. G.0582.14 awarded to Eva Ceulemans, Peter Kuppens and Francis Tuerlinckx), by the Belgian Federal Science Policy within the framework of the Interuniversity Attraction Poles program (IAP/P7/06), and by the Research Council of KU Leuven (GOA/15/003).

Software: Matlab based software for the method as well as the simulation studies is available upon request to the first author.